MSA220 Statistical Learning for Big Data

Lecture 4

Rebecka Jörnsten

Mathematical Sciences
University of Gothenburg and Chalmers University of Technology

Prediction models

- y is the output, an $n \times 1$ vector
- X is the input (explanatories, independent variables....), an $n \times p$ matrix
- We want to build a predictive model or rule for y using X

What is a good model?

- Need to define a loss function to define "good"
- Usually squared error loss if y is continuous
- Usually 0-1 error loss if y is categorical

Predictive models

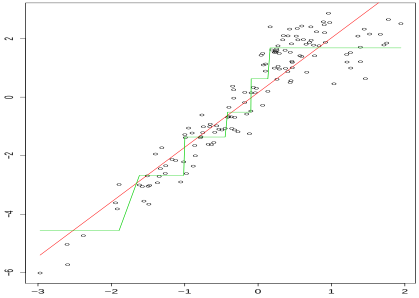
- $E(y - f(X))^2 = E_X[E_{y|X}[(y - f(X))^2 \mid X]]$
- We see that it is enough to minimize the above *pointwise* (for each $X = x$) so we focus on the inner expression
- $f(x) = \arg\min E_{y|X}[(y - f(x))^2 \mid X = x]$
- The minimizer is the *conditional mean* $\widehat{f(x)} = E(y \mid X = x)$
- For 0-1 loss the minimizer is the maximum probability class $\arg\max_c P(y = c \mid X = x)$.

The above conditional mean is called the *regression function*. It is the model that minimizes squared error loss.

How do we estimate $E(y \mid X = x)$?

- An intuitive approach is the approximate the conditional mean by the *local mean* defined by a neighborhood of observations $N(x)$ (e.g. k-nearest neighbors in X to observation x).
- Alternatively, we can parameterize the conditional mean via e.g. a linear model $E(y \mid X = x) = x'\beta$ or some other parametric form.

Predictive models



Red model: linear model. Green: local mean.

Bias-Variance Trade-off

How we choose to estimate the conditional mean determines how flexible/local or rigid/global we are in our modeling. Most methods allow us to choose from a spectrum of more or less local rules. We have already discussed that one needs to select tuning parameters for classification rules, e.g. the number of neighbors for kNN.

Depending on the tuning parameter value, a classification rule can be thought of as *local* or *global*.

local	global
use subset of data	use all data
flexible	more rigid
allow for complex boundaries or models	assume an underlying model for the data distribution
example: kNN with small k	example: discriminant analysis (multivariate normal data distribution)
example: Local average regression	example: linear regression model
need a lot of data to train on	requires less data in general

Discriminant Analysis

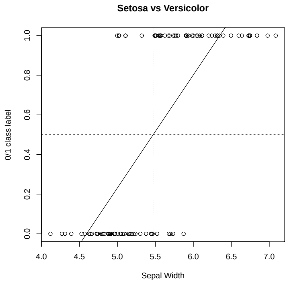
Discriminant analysis

CART has "problems" when the class boundaries are linear function in x-space. Such boundaries are poorly approximated by horizontal and vertical cuts. If your tree contains a lot of repeated splits on the same set of features (x1,x2,x1,x2,x1,x2) you can suspect that a linear boundary may be better. What CART is doing in this case is trying to approximate a linear boundary with a large number of small steps.
Discriminant analysis is a method that produces linear (and more complex) boundaries in x-space.
The underlying assumption that drives discriminant analysis methods is that the data distribution is *multivariate normal*.

0-1 REGRESSION

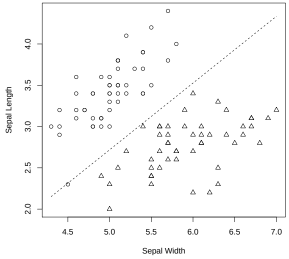
Before we dive into discriminant analysis we will look at a very simple rule based on regression.
If the data contains two classes, code these as a numerical variable y with values 0 and 1.
Let's consider the case with one x-variable. If you plot y versus x you can, hopefully, see a class separation in this scatter plot. Run regression of y on x and plot the fitted regression line. You can interpret this regression fit as an approximate estimate of the probability that the y equals 1 (the regression line is a linear model for $E(y \mid x)$ (conditional expectation of y given x) which is equal to $p(y = 1 \mid x)$ when y takes on only values 0 and 1.

Using the maximum posterior probability as your rule, the decision boundary equals the value for x when the regression line crosses $y = .5$: $\{x : \hat{y} = x\hat{\beta} = .5\}$ (here I use the notation for the fitted regression line, and the estimated regression coefficient $\hat{\beta}$).



0-1 REGRESSION

If you have more than one x-variable, you simply fit a multivariate regression model to the 0-1 data. The decision boundary can now be written as $\{x : x\hat{\beta} = .5\}$, where x and $\hat{\beta}$ are p-dimensional. This boundary expression is a linear equation system in x and can be solved for x-values such that the fitted value on the regression line equals 0.5.
Below is an example with 2 variables.



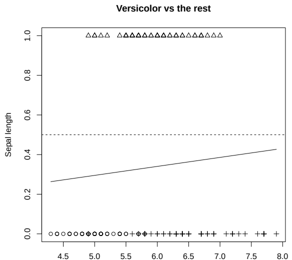
0-1 REGRESSION

So you see that naive 0-1 regression does a good job at constructing linear decision boundaries.
Any problems?

- The 0-1 regression line can produce negative values and values exceeding 1 which means it's a strange estimate for a probability. *Logistic regression* addresses this issue and the fact that one may want to weigh observations differently depending on how close to the boundary they are. See the Linear Models class for more info on Logistic regression

Problems?

- Masking. 0-1 regression may not work if you have more than 2 classes in your data. The strategy here is to perform many 0-1 regressions where each class takes a turn to be the 1-class and all the other observations are labeled 0. This may, however, result in the 1-class being hidden inside a cloud of 0-s and the regression line won't produce a sensible boundary. See figure below.
  This problem can sometimes be alleviated by running a polynomial regression model instead of a linear model.



Nearest centroid classifier

Another simple rule that is related to discriminant analysis is the nearest centroid classifier.
We compute

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{y_i = c} x_i$$

where N_c is the number of observations in class c. That is, we compute the mean, or centroid, of each class.
The rule is

$$\hat{c}(x) = \arg\min_c d(x, \hat{\mu}_c)$$

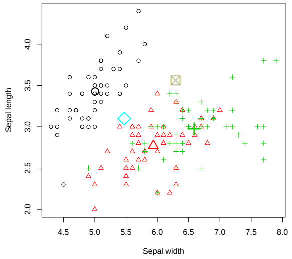
where $d(., .)$ is the distance between observation location x and the centroid μ. This is usually the euclidean distance

$$d(x, \hat{\mu}_c) = \| x - \hat{\mu}_c \|^2 = (x - \hat{\mu}_c)'(x - \hat{\mu}_c).$$

The rule is thus to allocate each observation to the class with the closest centroid.

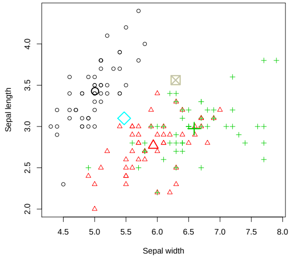
Nearest centroid classifier

Problems?
Nearest centroids ignores the variability of a class around its center and that this variability may be different for different classes and for different features.



For the iris data above, the black circle data (Setosa) exhibits much less variation than the other two classes. In addition, the black and red data sets are very tightly correlated in the two x-features, the green data less so.

Nearest centroid classifier



Consider the turqoise diamond. It is exactly halfway between the black and red class, but since the red class is much more spread out it seems more likely that this new observation belongs to the red class.

Consider the gray box. This is in fact closer to the green class center than the red class center. However, judging by the co-variation of the two *x*-features, the red class extends more in the direction of the gray box than the green class does so it is more likely this observation belongs to the red class.

Nearest centroid classifier

From the above examples it is clear that one needs to consider both *spread/scale* of a distribution (the amount of spread around a centroid) and the *shape* of the distribution (the correlation structure between the features) to form a good classification rule. This is what discriminant analysis adds to the table.

Discriminant analysis

General setup is the following;

- prior $\pi_c = p(y = c)$
- data distribution $p(x \mid y = c) \sim N(\mu_c, \Sigma_c)$ where μ_c is a p-dimensional vector and Σ_c is a p-by-p dimensional covariance matrix.

Discriminant analysis

The multivariate normal assumption leads to the following simple, intuitive parameter estimates:

- $\hat{\pi}_c = N_c/N$, where $N_c = \sum_i 1\{y_i = c\}$ is the number of observations in class c.
- $\hat{\mu}_c = \frac{1}{N_c} \sum_{y_i=c} x_i$
- $\hat{\Sigma}_c = \sum_{y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)'/(N_c - 1)$

This is quite a large number of parameters...: $(C-1)$ for $\hat{\pi}$ (not C since the πs add to 1), $p \times C$ mean parameters, and $p(p+1) \times C/2$ covariance parameters (since they're symmetric). As the dimensionality of the problem grows (p) the number of parameters grows quickly, especially due to the covariance matrices.

Linear discriminant analysis

The solution to this problem is to try to simplify the modeling assumption somewhat: $\Sigma_c = \Sigma$, same correlation structure between the features for all classes.

- Realistic? Think about the heart disease data. Do you think ldl-level and bmi are correlated the same way for healthy patients and patients with heart disease?
- The assumption may not be realistic BUT in statistics you always have to worry about the flexible methods suffering from poor estimation and thus leading to a bad classifier. Here, the approximation of equal correlation may be "safer" than trying to build a very complex method with many parameters on noisy data or a data with a small sample size.
- Under this assumption you get $\hat{\Sigma}$ from a *pooled* estimate.

$$\hat{\Sigma} = \sum_{c=1}^{C} \sum_{y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)'/(N-C) = \sum_{c=1}^{C} \hat{\Sigma}_c \frac{N_c - 1}{N - C},$$

a weighted average of covariance estimates from each individual class.

Other variants of discriminant analysis

You can make even more simplifying assumptions:

- $\Sigma_c = \Lambda_c$, diagonal matrix. You ignore the correlations between features. (DQDA) (Naive Bayes)
- $\Sigma_c = \Sigma = \Lambda$, diagonal matrix. You ignore correlations AND make the feature variance the same for all classes. (DLDA)
- $\Sigma_c = \Sigma = \sigma^2 I$, nearest centroid. Here you ignore all differences between classes and features in terms of variance and ignore feature correlations.

Building the classifier

We define the boundary between two classes, l and c, at the x-locations where the posterior probabilities are equal:

$$\{x : \pi_l p(x \mid y = l) = \pi_c p(x \mid y = c)\}$$

Equivalently, we can write this on a log-scale as

$$\{x : \log \frac{p(x \mid y = c)}{p(x \mid y = l)} + \log \frac{\pi_c}{\pi_l} = 0\}$$

Let's plug in the multivariate normal data distribution into this expression.

Building the classifier

$$p(x \mid y = c) = \frac{1}{(2\pi)^(p/2) \mid \Sigma_c \mid} exp(-\frac{1}{2}(x - \mu_c)'\Sigma_c^{-1}(x - \mu_c))$$

Taking logs

$$\log p(x \mid y = c) = -\frac{1}{2}(x - \mu_c)'\Sigma_c^{-1}(x - \mu_c) - \frac{1}{2}\log \mid \Sigma_c \mid - \frac{p}{2}\log 2\pi$$

and taking the log-ratio of the data-distributions we get

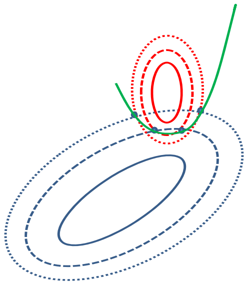
$$\log \frac{p(x \mid y = c)}{p(x \mid y = l)} = -\frac{1}{2}(x - \mu_c)'\Sigma_c(x - \mu_c) +$$

$$-\frac{1}{2}\log \mid \Sigma_c \mid + \frac{1}{2}(x - \mu_l)'\Sigma_c(x - \mu_l) + \frac{1}{2}\log \mid \Sigma_l \mid$$

Notice that this is *quadratic* in x, so class boundaries are quadratic curves in x-space.

Building the classifier

To draw these boundaries, simply look for points in x-space where the posterior distributions have the same value in two classes. I have illustrated that below with two classes and different line types corresponding to the contours for 90, 95 and 99 percent of the probability mass.



Building a linear classifier

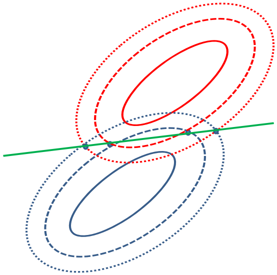
If you plug in the simplifying assumption that $\Sigma_c = \Sigma$ in the class boundary expression the quadratic terms in x cancel out and we get

$$\log \frac{p(x \mid y = c)}{p(x \mid y = l)} = -\frac{1}{2}(\mu_c + \mu_l)\Sigma^{-1}(\mu_c - \mu_l) + x\Sigma^{-1}(\mu_c - \mu_l) = 0$$

Notice that this is *linear* in x.

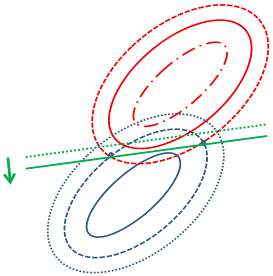
Building a linear classifier

To draw these boundaries, simply look for points in x-space where the posterior distributions have the same value in two classes. I have illustrated that below with two classes and different line types corresponding to the contours for 90, 95 and 99 percent of the probability mass.



Building a linear classifier

So what is the role of the prior? The prior will simply shift the contours of the data distribution center-outward if it increases, resulting in intersections with other class distribution contours further away from the distribution with a higher prior.



Discriminant analysis - rule

Let's focus on the rule instead of the boundary for a moment. The rule, as before, is the maximum posterior allocation. Here,

$$\hat{c}(x) = \arg\max_c \delta_c(x)$$

where

$$\delta_c(x) = \log \pi_c - \frac{1}{2}(x - \mu_c)' \Sigma_c^{-1} (x - \mu_c) - \frac{1}{2} \log \mid \Sigma_c \mid$$

Discriminant analysis - rule

If we consider the special case $\Sigma_c = \Sigma = \sigma^2 I$ (equal noise level for all features in all classes)

$$\delta_c(x) = \log \pi_c - \frac{1}{2\sigma^2}(x - \mu_c)'(x - \mu_c) + \ \mathit{constant},$$

i.e., the nearest centroid classifier, adjusted for the prior.

Discriminant analysis - rule

If we consider the special case $\Sigma_c = \Sigma$ (equal noise level and feature correlation structure in all classes)

$$\delta_c(x) = \log \pi_c - \frac{1}{2\sigma^2}(x-\mu_c)'\Sigma^{-1}(x-\mu_c) + \ \textit{constant} =$$

$$= \log \pi_c - \frac{1}{2}(\Sigma^{-1/2}(x-\mu_c))'(\Sigma^{-1/2}(x-\mu_c)) + \ \textit{constant}$$

The matrix $\Sigma^{-1/2}$ is the "square root" matrix, meaning the $\Sigma^{-1/2}\Sigma^{-1/2} = \Sigma^{-1}$. Writing the expression this way has the following benefit: you know that $Cov(X) = \Sigma$. If you apply the transformation $\Sigma^{-1/2}$ to the data the resulting covariance is $V(\Sigma^{-1/2}\ X) = \Sigma^{-1/2}\Sigma\Sigma^{-1/2} = I$. That is, the transformation $\Sigma^{-1/2}$ serves the purpose of decorrelating and standardizing the data. This is called *sphering* the data and moves the classification problem into a new coordinate system.

Discriminant analysis - rule

We can write

$$\log \pi_c - \frac{1}{2}(\Sigma^{-1/2}(x-\mu_c))'(\Sigma^{-1/2}(x-\mu_c)) + \textit{ constant} =$$

$$= \log \pi_c - \frac{1}{2}(\tilde{x}-\tilde{\mu}_c)'(\tilde{x}-\tilde{\mu}_c) + \textit{ constant} =$$

$$= \log \pi_c - \frac{1}{2}\sum_{j=1}^{p}(\tilde{x}_j-\tilde{\mu}_{cj})^2 + \textit{ constant}$$

where we define $\tilde{x} = \Sigma^{-1/2}x$ and $\tilde{\mu}_c = \Sigma^{-1/2}\mu_c$.
Notice that this is just the nearest centroid classifier! So LDA is nearest centroid in a new coordinate system formed by rotating and scaling the data x with respect to the covariance structure of x within each class.

Cautionary remarks

One take-home message from this lecture is that a key component in LDA is the inverse of Σ, the within-class covariance matrix. Problem?

- Σ may be very difficult to invert, numerically unstable, if the sample size n is not much bigger than the data dimension p.
- If some of the x-features are highly correlated, then the matrix Σ is also difficult to invert since it is near singular.
- Special case when $n < p$ or some x-s are perfectly correlated, the inverse of Σ does not exist. This is less of a worry since most programs will warn you about this. When we are *near* singular, that's when you need to pay attention. So correlated xs, high-dimensional data and small sample sizes are all situations when LDA can fail due to the poor performance of the inverse of Σ.

Cautionary remarks

There are some fixes we can consider. *Penalized discriminant analysis, PDA* is a numerical fix you may recognize from linear algebra class: we use $(\Sigma + \lambda I)^{-1}$ instead of Σ^{-1}. When Σ is near singular, adding a small values λ to the diagonal stabilizes the inverse operation.
In the high-dimensional case we will also consider *sparse* matrix inversion techniques, essentially restricting which values in the inverse are non-zero.
You can of course always reduce the number of features to consider, via manual selection, pre-screening or using principal components. We will come back to this in the high-dimensional part of the course.

Mini2

Pick 2-3 data sets and compare classification performance for

- kNN
- logistic/multinomial
- Variants of discriminant analysis
- CART
- RandomForest
- Your own, or using available R packages

You will also compare how these methods can be improved via *Ensemble methods*.
There are ensemble R packages available (just google "Ensemble Classifier R package") - BUT it might not be a bad idea to code this up yourself (even easier since some packages have a steep learning curve). In the demo, I use the caret package for which there is an ensemble extension (caretEnsemble) - but this refers to combining different classifiers NOT bagging.

Ensemble methods

An Ensemble Method:

1. Sample data with or without replacement
2. Learn your rule
3. Repeat steps 1-2 B times
4. On test data, use majority vote decision from the B rules

The above describes "Bagging" - bootstrap aggregating. Ensemble methods is a broader term than this (more later). What's the idea? The motivation behind RandomForest underpins this; if a method is unstable (meaning high variance estimate), "averaging" out many estimates suppresses the variance. However, *stable* rules do not generally benefit from Bagging. The more global a rule is, the more stable.

Mini2: For the 2-3 data sets, which method is best? Can you think of why that is? Did Bagging improve performance for kNN, logistic or discriminant methods? There is also an approach called *stacking*, where different classifiers are combined to provide an ensemble rule. Try this out on your data sets.